R for Travel Demand Modelers

Jeremy Raw

Federal Highway Administration

February 17, 2011

Overview

The first presentation (this one) talks about what R is

Overview

The first presentation (this one) talks about what R is

The second presentation discusses how R has been used for travel modeling

Overview

The first presentation (this one) talks about what R is

The second presentation discusses how R has been used for travel modeling

This will be a quick overview:

Overview

The first presentation (this one) talks about what R is

The second presentation discusses how R has been used for travel modeling

This will be a quick overview:

"Curiosity Building" rather than "Capacity Building"

Overview

Overview

Overview

1. What R is, and why it matters
2. A Quick Tour of R
3. Trying R yourself

What is R?

- Free Software Environment for Data Analysis and Programming

What is R?

- Free Software Environment for Data Analysis and Programming
  - Infoworld 2010 BOSS Award
    (Best Open Source Software)

What is R?

- Free Software Environment for Data Analysis and Programming
  - Infoworld 2010 BOSS Award
    (Best Open Source Software)
- R is an implementation of the S language

What is R?

- Free Software Environment for Data Analysis and Programming
  - Infoworld 2010 BOSS Award
    (Best Open Source Software)
- R is an implementation of the S language
- S was invented by John Chambers, who wrote

What is R?

- Free Software Environment for Data Analysis and Programming
  - Infoworld 2010 BOSS Award (Best Open Source Software)
- R is an implementation of the S language
- S was invented by John Chambers, who wrote

The goal of R is "to turn ideas into software, quickly and faithfully"

R "encourages you to slide into programming, perhaps without noticing"

Why is R called an 'Environment'?

"The term 'environment' is intended to characterize R as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

"Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented."

From the R website (`http://www.r-project.org`)

Why is R called an 'Environment'?

The R "environment" includes

- A Graphical User Interface, plus command line programs for running R commands from a text file

Why is R called an 'Environment'?

The R "environment" includes

- A Graphical User Interface, plus command line programs for running R commands from a text file
- A powerful, expressive and complete language for describing data manipulations ("S")

Why is R called an 'Environment'?

The R "environment" includes

- A Graphical User Interface, plus command line programs for running R commands from a text file
- A powerful, expressive and complete language for describing data manipulations ("S")
- Access to thousands of well-tested user-contributed packages, and tools to write your own

Why is R called an 'Environment'?

The R "environment" includes

- A Graphical User Interface, plus command line programs for running R commands from a text file
- A powerful, expressive and complete language for describing data manipulations ("S")
- Access to thousands of well-tested user-contributed packages, and tools to write your own
- Extensive documentation

Why is R called an 'Environment'?

The R "environment" includes

- A Graphical User Interface, plus command line programs for running R commands from a text file
- A powerful, expressive and complete language for describing data manipulations ("S")
- Access to thousands of well-tested user-contributed packages, and tools to write your own
- Extensive documentation
- Support for Windows (32- and 64-bit), Macintosh, and Linux

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated suites of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

Why Should I be Interested in R?

R is

- Interactive
- Fast
- Flexible
- Expressive
- Not Too Hard to Learn

Plus R is portable and scalable:

- You don't need administrator rights to run
- Commands run identically on all operating systems
- You can use it on your desktop
- You can write, save and distribute sophisticated sets of data and commands for others to use

A *Really* Quick Tour of R

Caution! We're going to move very fast

A *Really* Quick Tour of R

Caution! We're going to move very fast

But the R code from this presentation is available...

A *Really* Quick Tour of R

Caution! We're going to move very fast

But the R code from this presentation is available...

... and the presentation itself was written in R (and LaTeX)

More on that later.

Getting Started with R

The usual way of interacting with R is through its GUI.
(Graphical User Interface)

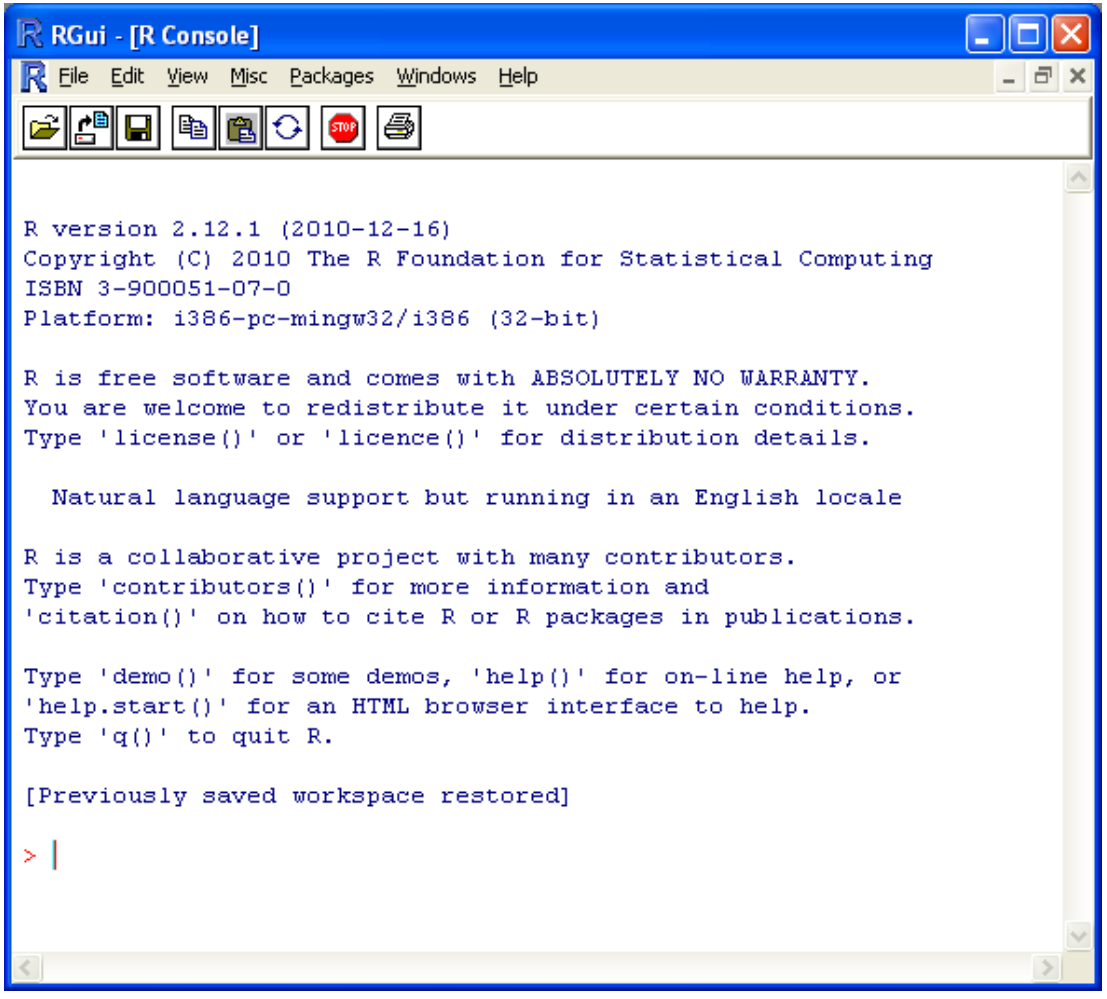
Getting Started with R

The usual way of interacting with R is through its GUI.
(Graphical User Interface)

The program `Rgui` opens a window into which you type commands.

Getting Started with R

The RGui window looks like this:



Handy R Commands

The most important R commands are these:

- `help.start()` which launches the help system
- `q()` which exits the program

Handy R Commands

The most important R commands are these:

- `help.start()` which launches the help system
- `q()` which exits the program

When you quit, you will be prompted to save the workspace.

The workspace is reloaded when you start R again.

Handy R Commands

The most important R commands are these:

- `help.start()` which launches the help system
- `q()` which exits the program

When you quit, you will be prompted to save the workspace.

The workspace is reloaded when you start R again.

Two files are saved:

1. `.Rdata`: all the data and functions you created
2. `.Rhistory`: a list of all the commands you entered

Handy R Commands

The most important R commands are these:

- `help.start()` which launches the help system
- `q()` which exits the program

When you quit, you will be prompted to save the workspace.

The workspace is reloaded when you start R again.

Two files are saved:

1. `.Rdata`: all the data and functions you created
2. `.Rhistory`: a list of all the commands you entered
    - Commands in `.Rhistory` may be copied to other script files for later use

Handy R Commands

The most important R commands are these:

- `help.start()` which launches the help system
- `q()` which exits the program

When you quit, you will be prompted to save the workspace.

The workspace is reloaded when you start R again.

Two files are saved:

1. `.Rdata`: all the data and functions you created
2. `.Rhistory`: a list of all the commands you entered
   - Commands in `.Rhistory` may be copied to other script files for later use
   - Run scripts with `Rterm` or `Rscript`

Handy R Commands

The most important R commands are these:

- `help.start()` which launches the help system
- `q()` which exits the program

When you quit, you will be prompted to save the workspace.

The workspace is reloaded when you start R again.

Two files are saved:

1. `.Rdata`: all the data and functions you created
2. `.Rhistory`: a list of all the commands you entered
   - Commands in `.Rhistory` may be copied to other script files for later use
   - Run scripts with `Rterm` or `Rscript`
   - Or use the `source()` function in `RGui`

Recap

To do useful things in R:

- Start RGui

Recap

To do useful things in R:

- Start `RGui`
- Enter `help.start()` to figure out what to do

Recap

To do useful things in R:

- Start `RGui`
- Enter `help.start()` to figure out what to do
- Enter other commands

Recap

To do useful things in R:

- Start `RGui`
- Enter `help.start()` to figure out what to do
- Enter other commands
- Use `q()` to quit

Approaching the R Language

For the rest of this tour of R we'll show some examples of R code and output.

Approaching the R Language

For the rest of this tour of R we'll show some examples of R code and output.

Reminder: The goal here is not to teach you R but to show you what it can do.

Welcome to R

R understands

- Integers

```
> 2
```

```
[1] 2
```

Welcome to R

R understands

- Integers

  ```
  > 2
  [1] 2
  ```

- Floating point numbers

  ```
  > 3.1415924
  [1] 3.141592
  ```

Welcome to R

R understands

- Integers

```
> 2
[1] 2
```

- Floating point numbers

```
> 3.1415924
[1] 3.141592
```

- Character strings

```
> "Hello, world"
[1] "Hello, world"
```

R Functions

R is a "functional language" – functions do it all

R Functions

R is a "functional language" – functions do it all

Here is a function that prints an integer:

```
> print(2)
[1] 2
```

R Functions

R is a "functional language" – functions do it all

Here is a function that prints an integer:

```
> print(2)
```

```
[1] 2
```

And here is a function that reports the length of a vector:

```
> length(2)  # One number is still a vector
```

```
[1] 1
```

R Functions

R is a "functional language" – functions do it all

Here is a function that prints an integer:

```
> print(2)
```

```
[1] 2
```

And here is a function that reports the length of a vector:

```
> length(2)  # One number is still a vector
```

```
[1] 1
```

Here is a function to get help about a function:

```
> help("length")
```

R Vectors

Here is a function to create a vector:

```
> c(1,2,3,4)  # concatenate values
[1] 1 2 3 4
> 1:4
[1] 1 2 3 4
```

R Vectors

Here is a function to create a vector:

```
> c(1,2,3,4)  # concatenate values
[1] 1 2 3 4
> 1:4
[1] 1 2 3 4
```

- Vectors are the key to fast processing in R

R Vectors

Here is a function to create a vector:

```
> c(1,2,3,4)  # concatenate values
[1] 1 2 3 4
> 1:4
[1] 1 2 3 4
```

- Vectors are the key to fast processing in R

Multiplication by a scalar

```
> c(1, 2, 3, 4) * 2
[1] 2 4 6 8
```

R Vectors

Here is a function to create a vector:

```
> c(1,2,3,4)  # concatenate values
[1] 1 2 3 4
> 1:4
[1] 1 2 3 4
```

- Vectors are the key to fast processing in R

Multiplication by a scalar

```
> c(1, 2, 3, 4) * 2
[1] 2 4 6 8
```

Element by Element Division (R also does linear algebra)

```
> c(8, 27, 64)/c(2, 3, 4)
[1]  4  9 16
```

Statistics

```
> runif(4)  # 4 draws from the interval [0,1]
[1] 0.28290481 0.07866804 0.48956254 0.19959806
```

Statistics

```
> runif(4)  # 4 draws from the interval [0,1]
[1] 0.28290481 0.07866804 0.48956254 0.19959806
> rnorm(4)  # 4 draws from N(0,1)
[1] -0.0546782 -0.9527661  0.5314582  1.4766443
```

Statistics

```
> runif(4)  # 4 draws from the interval [0,1]
[1] 0.28290481 0.07866804 0.48956254 0.19959806
> rnorm(4)  # 4 draws from N(0,1)
[1] -0.0546782 -0.9527661  0.5314582  1.4766443
> dnorm(1.8,mean=2,sd=0.5) # normal density
[1] 0.7365403
```

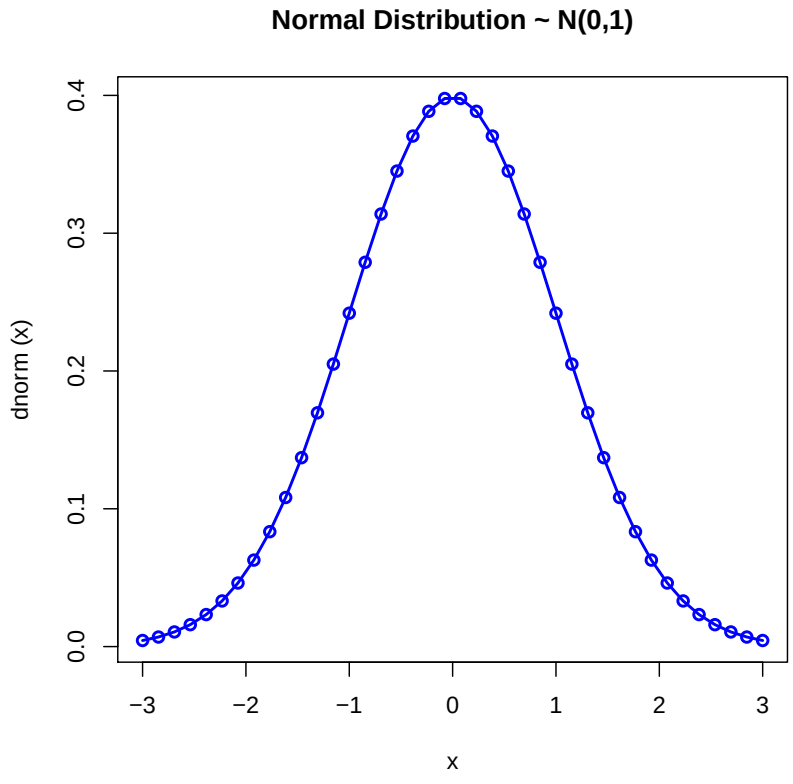
Statistics

```
> runif(4)  # 4 draws from the interval [0,1]
[1] 0.28290481 0.07866804 0.48956254 0.19959806
> rnorm(4)  # 4 draws from N(0,1)
[1] -0.0546782 -0.9527661  0.5314582  1.4766443
> dnorm(1.8,mean=2,sd=0.5) # normal density
[1] 0.7365403
> sample(5)     # 5 integers in random order
[1] 2 3 5 1 4
> sample(56,5)  # 5 random integers between 1 and 56
[1] 26 39 19 22  8
> sample(46,1)  # 1 random integer between 1 and 46
[1] 25
```

Plotting

Here is a graph of a normal distribution N(0,1)

```
> plot(dnorm,from=-3,to=3,n=40,
+       col="blue",type='o',lwd=2,
+       main='Normal Distribution ~ N(0,1)')
```



Variables

The basic R way to make a variable::

```
> seq4 <- 1:4
```

Variables

The basic R way to make a variable::

```
> seq4 <- 1:4
```

But you can also do it this way:

```
> 1:4 -> seq4.a
```

Variables

The basic R way to make a variable::

```
> seq4 <- 1:4
```

But you can also do it this way:

```
> 1:4 -> seq4.a
```

And, if you must, even this works:

```
> seq4.eq = 1:4
```

Variables

The basic R way to make a variable::

```
> seq4 <- 1:4
```

But you can also do it this way:

```
> 1:4 -> seq4.a
```

And, if you must, even this works:

```
> seq4.eq = 1:4
```

Find out what objects you have:

```
> objects()
[1] "seq4"    "seq4.a"  "seq4.eq"
```

Variables

The basic R way to make a variable::

```
> seq4 <- 1:4
```

But you can also do it this way:

```
> 1:4 -> seq4.a
```

And, if you must, even this works:

```
> seq4.eq = 1:4
```

Find out what objects you have:

```
> objects()
```

```
[1] "seq4"    "seq4.a"  "seq4.eq"
```

Using a Unix-like alternative:

```
> ls()
```

```
[1] "seq4"    "seq4.a"  "seq4.eq"
```

Indexing Vectors

Indexing a vector extracts some of its elements

Indexing Vectors

Indexing a vector extracts some of its elements

```
> v <- c('A','B','C','D')
> v[3]        # A single element
```

```
[1] "C"
```

Indexing Vectors

Indexing a vector extracts some of its elements

```
> v <- c('A','B','C','D')
> v[3]        # A single element
```

```
[1] "C"
```

```
> v[1:2]      # A pair of elements
```

```
[1] "A" "B"
```

Indexing Vectors

Indexing a vector extracts some of its elements

```
> v <- c('A','B','C','D')
> v[3]       # A single element
[1] "C"
> v[1:2]     # A pair of elements
[1] "A" "B"
> idx <- c(1,4)
> v[idx] # Using a vector to index a vector
[1] "A" "D"
```

Indexing Vectors

Indexing a vector extracts some of its elements

```
> v <- c('A','B','C','D')
> v[3]       # A single element

[1] "C"

> v[1:2]     # A pair of elements

[1] "A" "B"

> idx <- c(1,4)
> v[idx] # Using a vector to index a vector

[1] "A" "D"

> v[5]       # Oops, out of bounds!

[1] NA
```

Indexing Vectors

But what about this?

```
> v <- c("A", "B", "C", "D")
> v[c(1, 2, 1, 3, 1, 2)]
```

Indexing Vectors

But what about this?

```
> v <- c("A", "B", "C", "D")
> v[c(1, 2, 1, 3, 1, 2)]
```

```
[1] "A" "B" "A" "C" "A" "B"
```

Indexing Vectors

But what about this?

```
> v <- c("A", "B", "C", "D")
> v[c(1, 2, 1, 3, 1, 2)]
```

```
[1] "A" "B" "A" "C" "A" "B"
```

Categorical data is stored in a "factor"

```
> f <- factor(c("HBO", "HBO", "HBW", "NHB", "HBW"))
> f
```

```
[1] HBO HBO HBW NHB HBW
Levels: HBO HBW NHB
```

```
> as.integer(f)
```

```
[1] 1 1 2 3 2
```

Factors are stored as integers, but are printed as levels

Matrices

A matrix is a two-dimensional vector.

```
> m <- matrix(1:9, ncol = 3)
> m
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrices

A matrix is a two-dimensional vector.

```
> m <- matrix(1:9, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> m <- matrix(1:9, ncol = 3, byrow = TRUE)
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Matrices

Some other matrix operations:

```
> t(m)
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
> diag(m)
```

```
[1] 1 5 9
```

```
> diag(m) <- 0
```

Matrices

Some other matrix operations:

```
> det(m)

[1] 180

> eigen(m)

$values
[1]  9.768639 -7.214613 -2.554027


$vectors
           [,1]       [,2]       [,3]
[1,] -0.3449886 -0.1757094 -0.7662572
[2,] -0.5897528 -0.5700571  0.5871849
[3,] -0.7301880  0.8025965  0.2608905
```

Matrix Indexing

Matrix indexing extends vector indexing.

```
> m[2:3,2:3] # Matrix subset
```

```
     [,1] [,2]
[1,]    0    6
[2,]    8    0
```

```
> m[,3]      # Vector (drops unit dimensions)
```

```
[1] 3 6 0
```

Matrix Indexing

Matrix indexing extends vector indexing.

```
> m[2,3]      # Scalar (drops both dimensions)
[1] 6
> m[2,3,drop=FALSE] # 1 x 1 matrix
     [,1]
[1,]    6
```

Lists

A `list` is a sequence of arbitrary objects (which can have names)

Lists

A `list` is a sequence of arbitrary objects (which can have names)

```
> a.list <- list(
+     Sequence=seq4,
+     Text=LETTERS[1:10]
+     )
> a.list

$Sequence
[1] 1 2 3 4


$Text
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

Object Structure

Find out what is in a list:

```
> names(a.list)
[1] "Sequence" "Text"
```

Object Structure

Find out what is in a list:

```
> names(a.list)
```

```
[1] "Sequence" "Text"
```

Explore an R object's detailed structure:

```
> str(a.list)
```

```
List of 2
 $ Sequence: int [1:4] 1 2 3 4
 $ Text    : chr [1:10] "A" "B" "C" "D" ...
```

Lists

Lists are indexed in various ways

Lists

Lists are indexed in various ways

```
> a.list[2]     # second element as list
```

```
$Text
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

Lists

Lists are indexed in various ways

```
> a.list[2]     # second element as list

$Text
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"

> a.list[[2]]  # second element as vector

 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

Lists

Lists are indexed in various ways

```
> a.list[2]     # second element as list

$Text
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"

> a.list[[2]]  # second element as vector

 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"

> a.list$Text  # List element by Name

 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

Lists

Lists are indexed in various ways

```
> a.list[2]     # second element as list

$Text
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"

> a.list[[2]]  # second element as vector

 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"

> a.list$Text  # List element by Name

 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"

> a.list[["Text"]]  # Also by name

 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

Data Frames

`data.frames` are used to store tables of data.

Data Frames

`data.frames` are used to store tables of data.

A `data.frame` is a list of vectors, all the same length.

```
> seq5 <- 2:5
> df <- data.frame(N=seq5,S=seq5^2,C=seq5^3)
```

Data Frames

`data.frames` are used to store tables of data.

A `data.frame` is a list of vectors, all the same length.

```
> seq5 <- 2:5
> df <- data.frame(N=seq5,S=seq5^2,C=seq5^3)

> names(df)    # The list of columns (or fields)

[1] "N" "S" "C"
```

Data Frames

`data.frames` are used to store tables of data.

A `data.frame` is a list of vectors, all the same length.

```
> seq5 <- 2:5
> df <- data.frame(N=seq5,S=seq5^2,C=seq5^3)

> names(df)    # The list of columns (or fields)

[1] "N" "S" "C"
```

The corresponding elements of each vector make up the rows

```
> df

  N  S   C
1 2  4   8
2 3  9  27
3 4 16  64
4 5 25 125
```

Indexing Data Frames

`data.frames` are indexed like matrices and lists

```
> df[2:3,2:3] # Subset
```

```
   S  C
2  9 27
3 16 64
```

```
> df[2]       # just the second column
```

```
   S
1  4
2  9
3 16
4 25
```

```
> df[[2]]     # second column as a vector
```

```
[1]  4  9 16 25
```

Indexing Data Frames

`data.frames` are indexed like matrices and lists

```
> df[2,]      # just the second row
```

```
  N S  C
2 3 9 27
```

Not a vector, since data.frames can have different types in each column

```
> df$S        # vector from 'S' column
```

```
[1]  4  9 16 25
```

```
> df[sample(nrow(df),1), ]  # random row
```

```
  N  S   C
4 5 25 125
```

Loading Data

- A `data.frame` is often created from data collected outside R.

Loading Data

- A `data.frame` is often created from data collected outside R.
- R supports text formats directly.

Loading Data

- A `data.frame` is often created from data collected outside `R`.
- `R` supports text formats directly.
  - Text data is imported with functions like `read.csv()`

Loading Data

- A `data.frame` is often created from data collected outside R.
- R supports text formats directly.
  - Text data is imported with functions like `read.csv()`
- Packages support data in other formats (e.g. DBF, SAS Export, SPSS, Excel)

Loading Data: An Example

2009 National Household Transportation Survey (NHTS) data:

```
> download.file(
+     "http://nhts.ornl.gov/2009/download/Ascii.zip",
+     "NHTS-2009-ASCII.zip"
+     )
> unzip("NHTS-2009-ASCII.zip",exdir="NHTS-2009")
> dir("NHTS-2009")  # show files in the directory

> hh  <- read.csv("NHTS-2009/hhv2pub.csv")
> veh <- read.csv("NHTS-2009/vehv2pub.csv")
> day <- read.csv("NHTS-2009/dayv2pub.csv")
> per <- read.csv("NHTS-2009/perv2pub.csv")
> save(per,hh,veh,day,file="NHTS.Rdata")

> load("NHTS.Rdata")   # reload later
```

But What Can R *Really* Do?

Let's look at some results and just zoom past the code.

But What Can R *Really* Do?

Let's look at some results and just zoom past the code.
But remember:

But What Can R *Really* Do?

Let's look at some results and just zoom past the code.
But remember:

- R supports "Reproducible Research" and "Literate Programming" (q.v.)

But What Can R *Really* Do?

Let's look at some results and just zoom past the code.
But remember:

- R supports "Reproducible Research" and "Literate Programming" (q.v.)
- This presentation itself was generated using R and other open source tools.

But What Can R *Really* Do?

Let's look at some results and just zoom past the code.
But remember:

- R supports "Reproducible Research" and "Literate Programming" (q.v.)
- This presentation itself was generated using R and other open source tools.
    - Get the presentation files from the file download area.

But What Can R *Really* Do?

Let's look at some results and just zoom past the code.
But remember:

- R supports "Reproducible Research" and "Literate Programming" (q.v.)
- This presentation itself was generated using R and other open source tools.
  - Get the presentation files from the file download area.
- You can study the code (and find all the mistakes)!

Trip Rates from NHTS

- Trips Per Person Per Day, by Purpose and Household Size

```
> print(trips.per.person.per.day)
```

	PURPOSE				
HHSIZE	HBO	HBSHOP	HBSOCREC	HBW	NHB
1	0.5053646	0.9330562	0.4571780	0.4104453	1.211096
2	1.1779582	1.7553935	0.9405527	0.9331598	2.408037
3	2.5857568	2.6211519	1.6863807	1.5631168	3.737891
4+	5.0528199	3.3497913	2.7294158	1.8686832	5.304584

DISCLAIMER:
This page shows how to use R, not how to analyze the NHTS.

Trip Rates from NHTS

- Trips Per Day, by Purpose

```
> print(mean.trips.per.day.by.purpose)

     HBO   HBSHOP HBSOCREC      HBW      NHB
2.089154 2.023070 1.328052 1.098461 2.938784
```

DISCLAIMER:
This page shows how to use R, not how to analyze the NHTS.

Trip Rates from NHTS

- Trips Per Day, All Households

```
> print(mean.trips.per.day)
[1] 9.477522
```

DISCLAIMER:
This page shows how to use R, not how to analyze the NHTS.

Trip Rates from NHTS

Here's how we did it:

```
> day.wt <- day[c("HOUSEID","WTTRDFIN",
+                 "TRIPPURP","TRVLCMIN")]
> hh.wt <- hh[c("HOUSEID","WTHHFIN","HHSIZE")]
> hh.wt$HHBIN <- with(hh.wt,{
+     cut(HHSIZE,
+         breaks=c(0,1,2,3,
+         max(HHSIZE)),
+         labels=c("1","2","3","4+"),
+         ordered_result=TRUE)
+     })
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Rates from NHTS

Here's how we did it:

```
> trips <- merge(day.wt,hh.wt,by="HOUSEID")
> trips$PURPOSE <- factor(trips$TRIPPURP,
+                                    exclude=c("-9"))
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Rates from NHTS

Here's how we did it:

```
> trips.person <- with(trips,{
+     aggregate(
+         trips["WTTRDFIN"],
+         by=list(PURPOSE=PURPOSE,HHBIN=HHBIN),
+         sum
+         )
+     })
> trips.person <- reshape(
+     trips.person,direction="wide",
+     idvar="HHBIN",timevar="PURPOSE",
+     v.names="WTTRDFIN"
+     )
> row.names(trips.person) <-
+         levels(trips.person$HHBIN)
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Rates from NHTS

Here's how we did it:

```
> trips.person.day <-
+  data.matrix(trips.person[,2:length(trips.person)]/365)
> dimnames(trips.person.day) <-
+    list(
+       HHSIZE=levels(trips.person$HHBIN),
+       PURPOSE=levels(trips$PURPOSE)
+       )
> trips.hh <- with(hh.wt,{
+     aggregate(
+         hh.wt["WTHHFIN"],
+         by=list(HHBIN=HHBIN),
+         sum
+         )
+     })
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Rates from NHTS

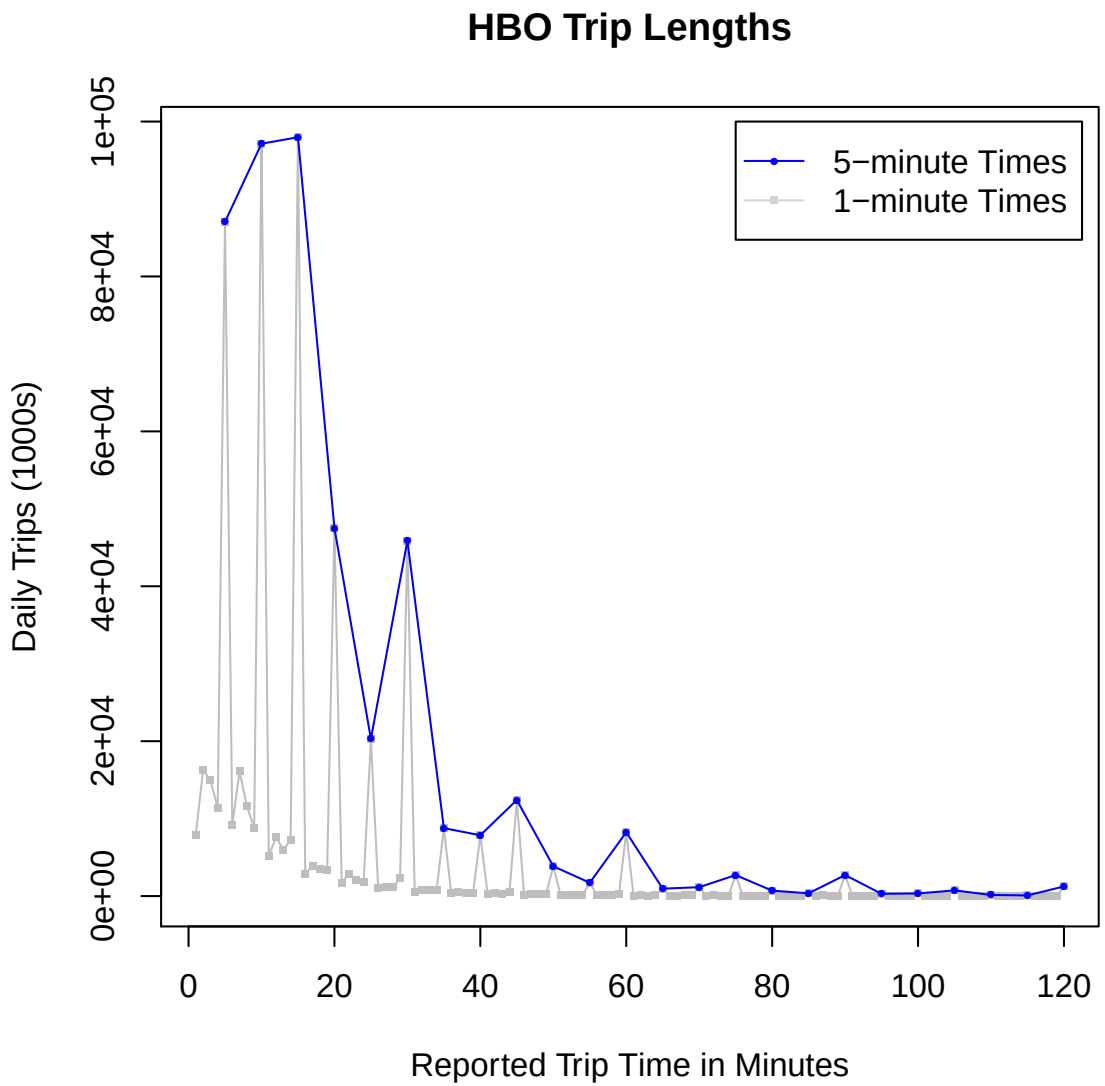
Here's how we did it:

```
> trips.per.person.per.day <-
+    sweep(trips.person.day,1,trips.hh$WTHHFIN,FUN="/")
> sum.trips.day <- colSums(trips.person.day)
> sum.hh <- sum(trips.hh$WTHHFIN)
> mean.trips.per.day.by.purpose <- sum.trips.day/sum.hh
> mean.trips.per.day <- sum(mean.trips.per.day.by.purpose)
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

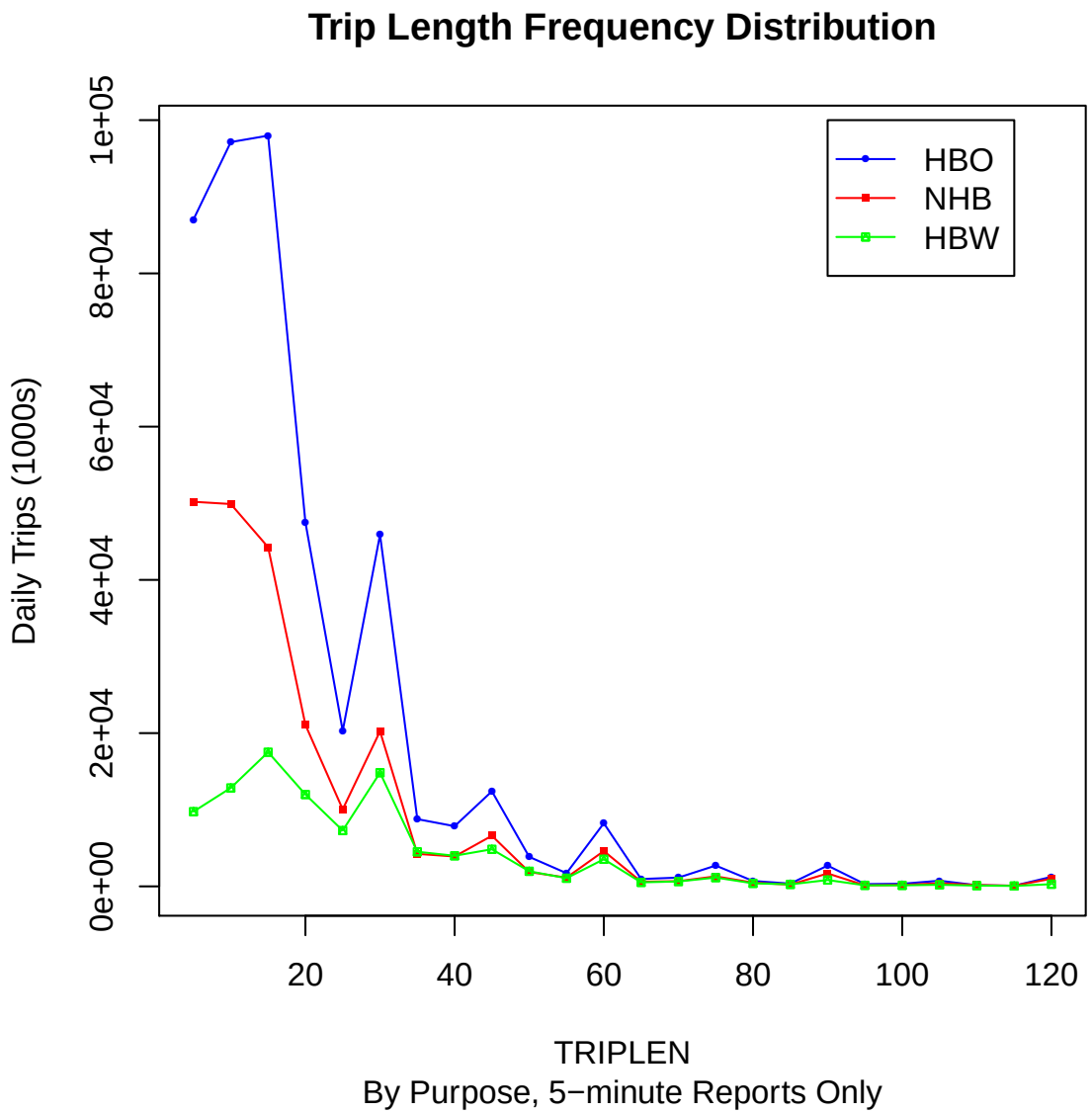
Trip Length Frequency Distributions

Trip Length Frequency Distributions from NHTS 2009



Trip Length Frequency Distributions

Trip Length Frequency Distributions from NHTS 2009



Trip Length Frequency Distributions

Here's how we did it:

```
> trips$SIMPPURP <- trips$PURPOSE
> recode <- which(
+   ! (   trips$PURPOSE %in% c("HBW","NHB"))
+           & !is.na(trips$PURPOSE )
+      )
> trips$SIMPPURP[recode] <- "HBO"
> trips$SIMPPURP <- factor(trips$SIMPPURP)
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Length Frequency Distributions

Here's how we did it:

```
> trips$TRIPLEN.1 <- as.integer(
+     cut( trips$TRVLCMIN,
+          breaks=seq(0,120,1),
+          ORDERED_RESULT=TRUE )
+     )
> trip.dist.1 <- with( trips, {
+     aggregate(
+         data.frame(Trips=WTTRDFIN/365000),
+         by=list(Purpose=SIMPPURP,TRIPLEN=TRIPLEN.1),
+         sum)
+     })
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Length Frequency Distributions

Here's how we did it:

```
> min.5 <- which(trips$TRVLCMIN%%5<0.1)
> trips[min.5,"TRIPLEN.5"] <- as.integer(
+     cut(
+         trips[min.5,"TRVLCMIN"],
+         breaks=seq(0,120,5),
+         ordered_result=TRUE,
+         )
+     ) * 5
> trip.dist.5 <- with( trips[min.5,], {
+     aggregate(
+         data.frame(Trips=WTTRDFIN/365000),
+         by=list(Purpose=SIMPPURP,TRIPLEN=TRIPLEN.5),
+         sum)
+     })
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Length Frequency Distributions

Here's how we did it:

```
> nhb.5 = which(trip.dist.5$Purpose=="NHB")
> hbw.5 = which(trip.dist.5$Purpose=="HBW")
> hbo.5 = which(trip.dist.5$Purpose=="HBO")
> hbo.1 = which(trip.dist.1$Purpose=="HBO")
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Length Frequency Distributions

Here's how we did it:

```
> plot(trip.dist.1[hbo.1,2:3],type='o',
+      col="gray",pch=15,cex=0.5,
+      main="HBO Trip Lengths",
+      xlab="Reported Trip Time in Minutes",
+   ylab="Daily Trips (1000s)")
> points(trip.dist.5[hbo.5,2:3],type='o',
+        col="blue",pch=16,cex=0.5)
> legend(75,100000,
+        c("5-minute Times","1-minute Times"),
+        col=c("blue","lightgray"),pch=c(16,15),
+        pt.cex=0.5,lty=1)
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

Trip Length Frequency Distributions

Here's how we did it:

```
> plot(trip.dist.5[hbo.5,2:3],type='o',
+      col="blue",pch=16,cex=0.5,
+      main="Trip Length Frequency Distribution",
+      sub="By Purpose, 5-minute Reports Only",
+      ylab="Daily Trips (1000s)")
> points(trip.dist.5[nhb.5,2:3],type='o',
+        col="red",pch=15,cex=0.5)
> points(trip.dist.5[hbw.5,2:3],type='o',
+        col="green",pch=14,cex=0.5)
> legend(90,100000,c("HBO","NHB","HBW"),
+        col=c("blue","red","green"),
+        pch=c(16,15,14),pt.cex=0.5,lty=1)
```

DISCLAIMER: This page shows how to use R, not how to analyze the NHTS.

R Packages

- All we have used so far is "Base R"

R Packages

- All we have used so far is "Base R"
- R is extended with "packages"

R Packages

- All we have used so far is "Base R"
- R is extended with "packages"
- A package is a set of functions and data, with documentation

Using Packages

- `library()` function to load functions from a package

Using Packages

- `library()` function to load functions from a package
- `data()` function to load data from a package

Using Packages

- `library()` function to load functions from a package
- `data()` function to load data from a package
- `help()` function (or `help.start()`) to load documentation from a package

Using Packages

- `library()` function to load functions from a package
- `data()` function to load data from a package
- `help()` function (or `help.start()`) to load documentation from a package

Most packages exist in compiled versions for all platforms

Navigating Packages

- The sheer number of R packages can be daunting

Navigating Packages

- The sheer number of R packages can be daunting
- You'll find packages you want from tutorials and other reading

Navigating Packages

- The sheer number of R packages can be daunting
- You'll find packages you want from tutorials and other reading
- "Task Views" tour packages for various subject areas
  - `http://cran.r-project.org/web/views`

Navigating Packages

- The sheer number of R packages can be daunting
- You'll find packages you want from tutorials and other reading
- "Task Views" tour packages for various subject areas
  - `http://cran.r-project.org/web/views`
- In addition to CRAN, there are several other important R Repositories:

  `http://www.bioconductor.org` Genomic research

  `http://r-forge.r-project.org` Development

  Look here for `travelr`, with functions for travel modeling

  `http://www.rforge.net` More Development

Interesting Packages

Spatial	`sp`, `maptools`, `rgdal`, `raster`, `spatstat`
Statistical	`MASS`, `nlme`, `nls`, `mlogit`, `survival`
Time Series	`forecast`, `tseries`
Bayesian Stats	`arm`, `bayesM`, `mcmc`, `sna`
GPS Analysis	`adehabitat`
Travel Modeling	`travelr` (from R-Forge)

Installing Packages

- Once you know what you want, installing packages is simple

Installing Packages

- Once you know what you want, installing packages is simple
- From the GUI menu, choose "Install Packages..."

Installing Packages

- Once you know what you want, installing packages is simple
- From the GUI menu, choose "Install Packages..."
- Or `install.packages()` like this:

```
> install.packages(c("rgdal", "sp"))
```

```
> # Non-standard repository:
> install.packages("travelr",
+         repos="http://r-forge.r-project.org")
```

Learning About R Packages

- In addition to help, many functions have working examples

```
> library(rgdal)
> help('readOGR')
> example('readOGR')
```

Learning About R Packages

- In addition to help, many functions have working examples

```
> library(rgdal)
> help('readOGR')
```

```
> example('readOGR')
```

- Some packages have "vignettes"

Learning About R Packages

- In addition to help, many functions have working examples

```
> library(rgdal)
> help('readOGR')

> example('readOGR')
```

- Some packages have "vignettes"
- A vignette is a PDF file, generated from R and LaTeX source code, just like this presentation

Learning About R Packages

- In addition to help, many functions have working examples

```
> library(rgdal)
> help('readOGR')

> example('readOGR')
```

- Some packages have "vignettes"
- A vignette is a PDF file, generated from R and LaTeX source code, just like this presentation
- The vignette serves two purposes:
  1. Shows how to use the package in common applications
  2. Delivers working code (used in the package build/test process)

Learning About R Packages

- In addition to help, many functions have working examples

```
> library(rgdal)
> help('readOGR')

> example('readOGR')
```

- Some packages have "vignettes"
- A vignette is a PDF file, generated from R and LaTeX source code, just like this presentation
- The vignette serves two purposes:
  1. Shows how to use the package in common applications
  2. Delivers working code (used in the package build/test process)
- Browse vignettes for installed packages:

```
> browseVignettes()
```

R Spatial Data

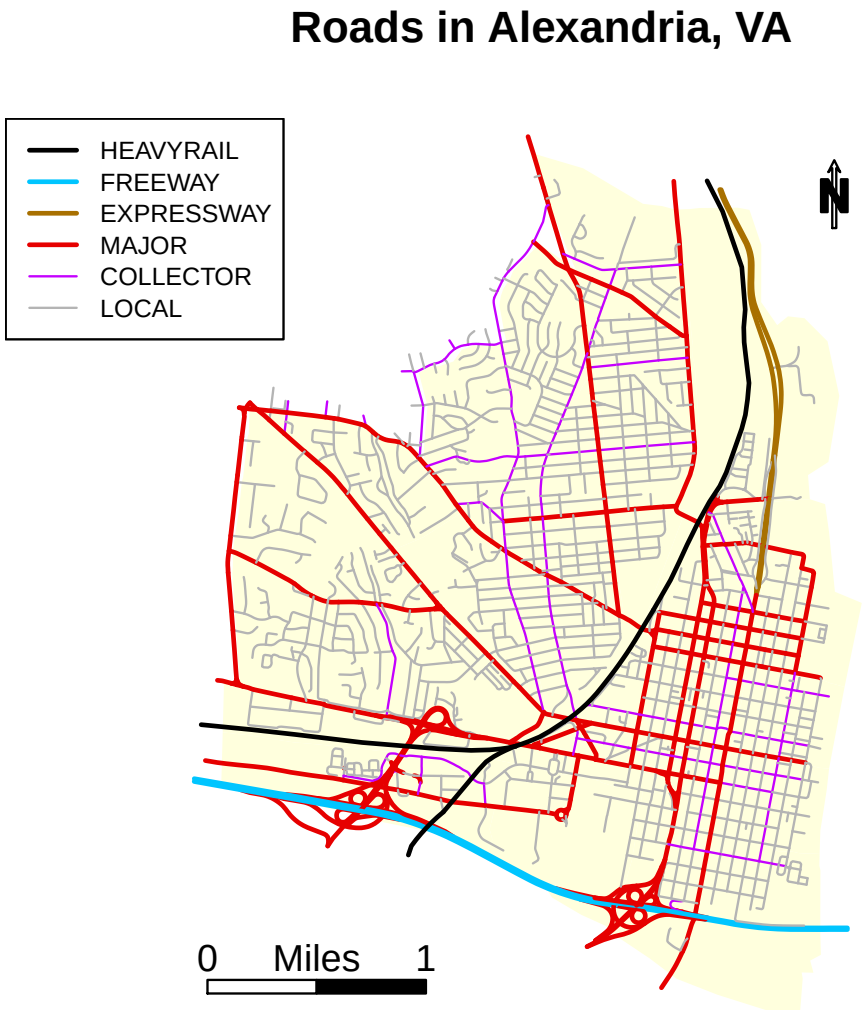
Here is an example of a map generated from R using one of the spatial analysis libraries

```
> library(sp)   # spatial data library
> load("alx.network.Rdata")   # sample data
```

(See the presentation source for the rest...)

R Spatial Data

Here is an example of a map generated from R using one of the spatial analysis libraries



Where to get R

- R is found on the Comprehensive R Archive Network
  - "CRAN", pronounced "See-Ran" or "Kran"

Where to get R

- R is found on the Comprehensive R Archive Network
  - "CRAN", pronounced "See-Ran" or "Kran"
- `http://cran.r-project.org/mirrors.html`

Where to get R

- R is found on the Comprehensive R Archive Network
  - "CRAN", pronounced "See-Ran" or "Kran"
- `http://cran.r-project.org/mirrors.html`
  - Pick a mirror site near you.

Some Learning Links R

- R is installed with a lot of documentation
  - see `help.start()`)

Some Learning Links R

- R is installed with a lot of documentation
  - see `help.start()`)
- More documentation through the R Project Website:
  - `http://www.r-project.org`

Some Learning Links R

- R is installed with a lot of documentation
  - see `help.start()`)
- More documentation through the R Project Website:
  - `http://www.r-project.org`
- The Tutorial Introduction can be read without installing R
  - `http://cran.r-project.org/doc/manuals/R-intro.html`

Accessing this Presentation

The file pod for this web presentation includes a zip archive with the following files:

Travel-Modeling-With-R.pdf Finished presentation
Travel-Modeling-WIth-R.R Just the R commands
Travel-Modeling-With-R.Rnw R + LaTeX source code
ReadMe.txt A text file explaining how to build the presentation
Rgui.png Screenshot of R GUI window
Rlogo.png R Logo
Sweave.sty Support file for building the Presentation
alx.network.Rdata Sample spatial data (R format)

If you want to build the presentation from the `.Rnw` file, you will need a working LaTeX installation. The MikTeX package (`http://miktex.org`) works well on Windows.

Otherwise, to try the code, you only need R itself, plus a working internet connection.